Final Project Milestone

Yang "Eddie" Chen, Yi Zhong, Yubo Tian

November 20, 2017

1. Motivation

Tennis matches are fun to watch because they are full of surprises. In the 2017 Stuttgart Open, Roger Federer, then an 18-time grand slam champion, lost to the world No. 302 player Tommy Haas. Federer lost the opening match at a grass-court tournament, a phenomenon that hadn't happened since 2002. How can we predict a rare loss like this? Based on past performance alone, any model would have predicted a Federer win in pre-game bets. This motivates us to apply machine learning to predict tennis matches in real time; in particular, we want to explore how well we can combine both historical performance data and real-time, just-happened set-by-set outcome, to achieve a better prediction. As a consequence, both tennis players and sports bettors can benefit from better predictions and new insights.

Tennis is an ideal candidate for a hierarchical model as a match consists of a sequence of sets, which in turn consist of a sequence of games, which in turn consist of a sequence of points...**TODO**: add more here to explain we start by looking at set-by-set [4]

This paper seeks to model men's professional singles matches...**TODO**: *add some introduction after we have better ideas*

2. Method

2.1. Feature Extraction

2.1.1. Feature Representation

A supervised machine learning algorithm requires a set of labeled examples for training. In our project, each training example combines both the historical aspect, which is the stats calculated from the past, and the real-time aspect, which is the immediately preceding set information. (Reminder that our project focuses on predicting match outcome form both the historical data and the real-time info). In terms of representation, the training example is made up of:

1. A vector of input features (X), describing the players and the match
2. A target value (y), indicating the match outcome

In any given singles match, we have two players in one game. We denote them as Player 1 and Player 2, and thus define the target value as $y = 1$ if Player 1 wins, 0 otherwise.

2.1.2. Symmetric Match & Set Feature Representation

As suggested by Sipko [4] and Clarke and Dyte [1], when considering the charateristics of *both* players in a given match, we are mainly interested in the *difference* between two players for our variables of interest. Take for example, the rank information between two players for a given match. We construct a single feature called $RANK_DIFF = RANK_1 - RANK_2$, where $RANK_1$ and $RANK_2$ are the ranks of players 1 and 2 respectively at the time of the match. We selected this design (as opposed to the alternative of including both players' info, such as both $RANK_1$ and $RANK_2$), because previous research has shown that the difference variables are predictive enough [4] [2], and this way we cut down the feature size by half as compared to the alternative. Reduced feature column count reduces the *variance* of the model. Moreover, using difference variables allows us to have a symmetric model: it will produce the same results even if we swap Player 1 and Player 2, and define $y = 1$ if Player 2 wins, 0 otherwise. Should we have two different variables such as $RANK_1$ and $RANK_2$ for an asymmetric model, the model might assign different weights to the two variables despite them referring to the same foundamental feature, only differing by the player labelling. Having a *symmetric* model eliminates the bias from that, as we only have one single difference variable for each feature now.

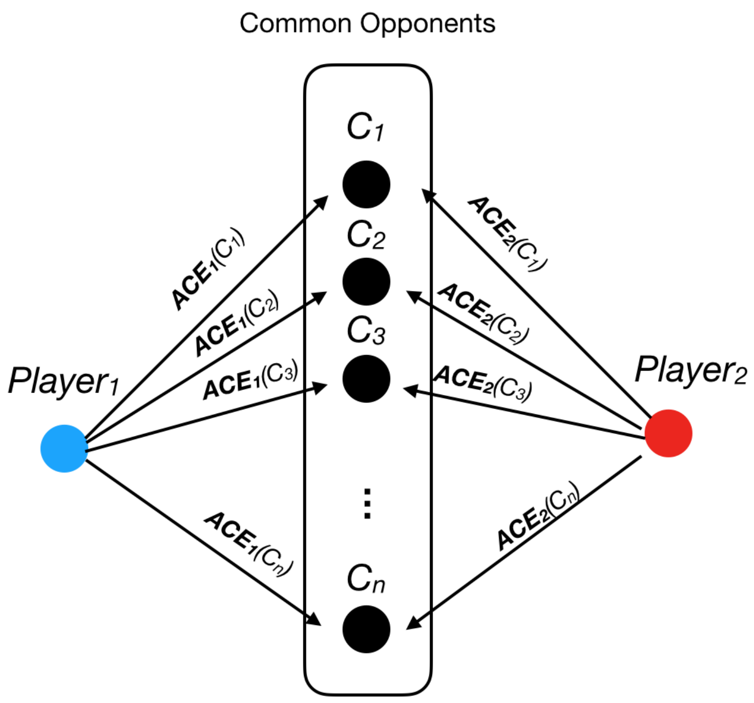
2.1.3. Hybridizing Match & Set Features

TODO

2.1.4. Common Opponent Model

We use a method proposed by Knottenbelt [3] to construct difference variables capturing player characteristics and past performance, by using their **common opponents** so as to achieve a fair comparison. Although this technique was developed for use in a hierarchical Markov model, Spiko [4] has shown that it adapts well in the land of machine learning algorithms and modeling.

Figure 2.1: Constructing the ACE feature using common opponents



To show how we construct the features, we first look for a set of **common opponents** between the two players of a given match. Here, **common opponents** are defined as other players that the two (Player 1 and Player 2) have faced in the past on their own. Once we have all the common opponents, we find both players' performance against the common opponent (C_i), and average across all the common opponents. Lastly, we find the difference between the two players' average stats against common opponents as the selected feature. Figure 2.1 shows an example of how we could construct the **ACE** feature (difference in average ace counts) against common opponents. As shown in Figure 2.1, common opponents are labeled as $C_1, C_2, ...C_n$. For Player $i \in \{1, 2\}$, $ACE_i(C_j)$ is his/her average count of aces in all matches against common opponent C_j. We then take an average of these values to get ACE_i for Player i:

$$ACE_i = \frac{\Sigma_{j=0}^{n} ACE_i(C_j)}{n}$$

Lastly, applying the idea of difference variables, we create the feature ACE by finding the difference in performance on this feature between Player 1 and Player 2:

$$ACE = ACE_1 - ACE_2$$

. We can apply the same technique on other features too. The limitation of this method is apparent: it only works well when players actually have common opponents have played games with them before the queried match date. Hence, we need to think about how to handle edge cases.

2.1.5. Edge Cases

We have run into 2 edge cases: (1) when at least one of the players is new; (2) when two players do not have any common opponents. When (1) happens, it is necessarily the case that it suffers from (2) no common opponents as well; but the reverse is not true. **TODO**: add in what we decided to do

2.2. Data Preparation

2.2.1. Data Source

We obtain match-level data from **TODO**

2.2.2. Data Cleaning & Transformation

TODO

3. Preliminary Experiments

3.1. Logistic Regression Model

3.2. Naive Bayes Classifier

4. Next Steps

5. Contributions

5.1. Yi Zhong

5.2. Yubo Tian

5.3. Yang "Eddie" Chen

- Researched and reviewed relevant literature for applicable ideas, such as Clarke and Dyte [1] and O'Malley [2] for difference variable, and Knottenbelt [3] for the common opponent model
- Implemented the Common Opponent Model to look up players stats from common opponents
- Generated ATP Match results for labels
- Typsetted ($\LaTeX$) the project milestone report

6. Conclusion

A. Appendix: Project Roadmap

- **11/13 - 11/19**: Data cleaning
- **11/20 - 11/26**: Preliminary experiements, building out more sophisticated ideas into models such as surface weighting, time decaying, fatigue, and injury
- **11/27 - 12/03**: Continue fine-tuning models
- **12/04 - 12/10**: Poster making
- **12/11 - 12/15**: Poster presentation; final project write-up

References

[1] Stephen R. Clarke and David Dyte. Using official ratings to simulate major tennis tournaments. *International Transactions in Operational Research*, 7(6):585 – 594, 2010.

[2] O'Malley A. James. Probability formulas and statistical analysis in tennis. *Journal of Quantitative Analysis in Sports*, 4(2):1–23, April 2008.

[3] William J. Knottenbelt, Demetris Spanias, and Agnieszka M. Madurska. A common-opponent stochastic model for predicting the outcome of professional tennis matches. *Computers and Mathematics with Applications*, 64(12):3820 – 3827, 2012. Theory and Practice of Stochastic Modeling.

[4] Michal Sipko. Machine learning for the prediction of professional tennis matches. Meng computing âĂŞ final year project, Imperial College London, June 2015. 15.